

HỌC VIỆN CÔNG NGHỆ BUƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1



BÁO CÁO BÀI TẬP

BÁO CÁO BÀI TẬP IOT VÀ ỨNG DỤNG

ĐỀ TÀI: XÂY DỰNG HỆ THỐNG CẢM BIẾN IOT

Sinh viên:

Bùi Ngọc Vũ

Mã sinh viên:

B22DCCN021

Giảng viên hướng dẫn:

TS. Nguyễn Quốc Uy

Hà Nội, 2025

MỤC LỤC

CHƯƠNG I: GIỚI THIỆU CHUNG.....	3
1.1. Bối cảnh và lý do chọn đề tài.....	3
1.2. Làm gì?.....	3
1.3. Phát triển như thế nào?.....	4
1.4. Phạm vi nghiên cứu và triển khai.....	5
1.5. Ý nghĩa và đóng góp.....	5
CHƯƠNG II. THIẾT KẾ HỆ THỐNG.....	6
2.1. Phân tích yêu cầu hệ thống.....	6
2.1.1. Phân tích yêu cầu hệ thống.....	6
2.1.2. Yêu cầu chức năng.....	6
2.2. Thiết kế phần cứng.....	7
2.2.1. Sơ đồ khối.....	7
2.2.2. Sơ đồ mạch.....	8
2.2.3. Sơ đồ thực tế.....	9
2.3. Thiết kế giao diện.....	9
2.3.1. Giao diện trang chủ.....	9
2.3.2. Giao diện dữ liệu cảm biến.....	10
2.3.3. Giao diện lịch sử hoạt động.....	11
2.3.4. Giao diện hồ sơ.....	12
2.4. Thiết kế cơ sở dữ liệu.....	13
2.4.1. Lược đồ ERB.....	13
2.5. Sequence Diagram.....	14
2.5.1. Sequence 1: HomeFrm hiển thị thiết bị, dữ liệu cảm biến, điều khiển ON/OFF.....	14
2.5.2. Sequence 2: Hiển thị dữ liệu cảm biến với tìm kiếm và lọc.....	17
CHƯƠNG III. XÂY DỰNG HỆ THỐNG.....	20
3.1. Cài đặt Arduino.....	20
3.2. Thiết kế hệ thống.....	25
3.3. Cấu trúc hệ thống.....	26
3.4. API.....	27
3.4.1. Lấy danh sách thiết bị.....	27
3.4.2. Gửi lệnh điều khiển.....	27
3.4.3. Tìm kiếm có điều kiện.....	27
3.5. Giao diện.....	27

3.5.1. Giao diện trang chủ.....	28
3.5.2. Giao diện dữ liệu cảm biến.....	29
3.5.3. Giao diện lịch sử hoạt động.....	30
3.5.4. Giao diện hồ sơ.....	31
CHƯƠNG IV. KẾT LUẬN.....	33
4.1. Tổng quan.....	33
4.2. Cải tiến trong tương lai.....	33
LỜI CẢM ƠN.....	35

CHƯƠNG I: GIỚI THIỆU CHUNG

1.1. Bối cảnh và lý do chọn đề tài

Trong những năm gần đây, cùng với sự phát triển mạnh mẽ của cuộc cách mạng công nghiệp 4.0, Internet of Things (IoT) đã và đang trở thành một trong những xu hướng công nghệ nổi bật. IoT cho phép kết nối hàng tỷ thiết bị, cảm biến, máy móc và hệ thống thông tin với nhau thông qua mạng Internet, tạo thành một hệ sinh thái thông minh. Nhờ đó, các hoạt động trong đời sống và sản xuất có thể được giám sát, điều khiển và tối ưu hóa một cách hiệu quả hơn.

Tại Việt Nam, ứng dụng IoT đang ngày càng được chú trọng và triển khai trong nhiều lĩnh vực khác nhau, từ nông nghiệp thông minh (giám sát độ ẩm, nhiệt độ đất, tự động tưới tiêu) cho đến quản lý giao thông, tòa nhà, nhà máy sản xuất hay hệ thống y tế. Trong đó, các hệ thống cảm biến IoT đóng vai trò nền tảng, bởi chúng đảm nhận nhiệm vụ thu thập dữ liệu từ môi trường thực tế và truyền về hệ thống xử lý trung tâm.

Thực tế hiện nay cho thấy nhu cầu về giám sát môi trường, quản lý năng lượng và điều khiển thiết bị điện tử từ xa ngày càng cao. Ví dụ, trong các hộ gia đình, việc kiểm soát nhiệt độ, độ ẩm hay ánh sáng có thể giúp tiết kiệm năng lượng, bảo vệ sức khỏe và nâng cao chất lượng cuộc sống. Trong sản xuất và công nghiệp, giám sát môi trường giúp kiểm soát chất lượng sản phẩm, đảm bảo an toàn lao động. Xuất phát từ nhu cầu đó, đề tài “*Xây dựng hệ thống cảm biến IoT*” được lựa chọn nhằm nghiên cứu, thiết kế và triển khai một mô hình IoT hoàn chỉnh có khả năng thu thập, lưu trữ, hiển thị và điều khiển dữ liệu cảm biến.

1.2. Làm gì?

Đề tài hướng đến việc xây dựng một hệ thống IoT có khả năng thực hiện các chức năng sau:

- Thu thập dữ liệu môi trường: Hệ thống sử dụng các cảm biến để đo các thông số cơ bản như nhiệt độ, độ ẩm và cường độ ánh sáng. Các dữ liệu này được thu nhận bởi vi điều khiển ESP32

- Truyền dữ liệu về máy chủ: Các dữ liệu sau khi được thu thập sẽ được gửi về máy chủ thông qua giao thức MQTT, đảm bảo tốc độ và độ tin cậy trong quá trình truyền tải.
- Lưu trữ dữ liệu: Dữ liệu được lưu trong cơ sở dữ liệu MySQL, giúp dễ dàng quản lý, truy vấn và phân tích.
- Hiển thị dữ liệu: Một giao diện web trực quan được xây dựng để hiển thị dữ liệu theo thời gian thực dưới dạng bảng và biểu đồ. Người dùng có thể dễ dàng theo dõi sự thay đổi của nhiệt độ, độ ẩm, ánh sáng theo từng khoảng thời gian.
- Điều khiển thiết bị từ xa: Ngoài chức năng giám sát, hệ thống còn hỗ trợ điều khiển các thiết bị điện tử (như đèn, quạt, điều hòa) từ xa thông qua giao diện web.

1.3. Phát triển như thế nào?

Hệ thống được phát triển dựa trên kiến trúc ba lớp điển hình của IoT, bao gồm:

- Lớp thiết bị (Device Layer): Sử dụng vi điều khiển ESP32 kết nối với cảm biến DHT11 (đo nhiệt độ và độ ẩm) và BH1750 (đo ánh sáng). Các cảm biến thu thập dữ liệu và gửi về ESP32 để xử lý sơ bộ trước khi truyền đi.
- Lớp mạng (Network Layer): Dữ liệu được truyền qua mạng WiFi và giao thức MQTT thông qua broker (HiveMQ). Giao thức này nhẹ, nhanh, phù hợp với các thiết bị IoT vốn hạn chế về tài nguyên.
- Lớp ứng dụng (Application Layer):
 - Backend: Xây dựng bằng Java Spring Boot, chịu trách nhiệm nhận dữ liệu từ MQTT, xử lý và lưu trữ vào MySQL. Backend cũng cung cấp các REST API và WebSocket để phục vụ frontend.
 - Frontend: Phát triển bằng HTML/CSS/JavaScript (hoặc ReactJS), sử dụng Chart.js để hiển thị biểu đồ dữ liệu cảm biến, đồng thời cung cấp giao diện trực quan cho người dùng điều khiển thiết bị.

Phương pháp phát triển của hệ thống được chia theo hướng phát triển từng chức năng. Quá trình triển khai bao gồm:

- Giai đoạn 1: Kết nối cảm biến và thu thập dữ liệu.
- Giai đoạn 2: Xây dựng backend để xử lý và lưu trữ dữ liệu.
- Giai đoạn 3: Phát triển frontend hiển thị dữ liệu theo thời gian thực.
- Giai đoạn 4: Kiểm thử hệ thống, đánh giá hiệu quả và tối ưu hóa.

1.4. Phạm vi nghiên cứu và triển khai

Do giới hạn về thời gian và nguồn lực, hệ thống được xây dựng ở quy mô thử nghiệm nhỏ, với một số lượng thiết bị giới hạn (2 thiết bị cảm biến: BHT11 và DH1750). Hệ thống tập trung vào các chức năng cốt lõi: thu thập, lưu trữ, hiển thị dữ liệu và điều khiển thiết bị. Các vấn đề nâng cao như bảo mật, mở rộng quy mô lớn, tích hợp trí tuệ nhân tạo để dự đoán xu hướng dữ liệu chưa được triển khai trong phạm vi đề tài, nhưng có thể được nghiên cứu và bổ sung trong tương lai.

1.5. Ý nghĩa và đóng góp

Đề tài “Xây dựng hệ thống cảm biến IoT” có ý nghĩa quan trọng cả về mặt học thuật lẫn thực tiễn:

- Về mặt học thuật: Đề tài giúp sinh viên tiếp cận toàn diện quy trình phát triển một hệ thống IoT, từ việc lập trình nhúng, quản lý cơ sở dữ liệu, phát triển backend đến xây dựng giao diện người dùng. Đây là cơ hội rèn luyện kỹ năng tích hợp phần cứng và phần mềm, áp dụng các công nghệ hiện đại vào một sản phẩm cụ thể.
- Về mặt thực tiễn: Hệ thống có thể áp dụng trong việc giám sát môi trường trong hộ gia đình, lớp học, phòng thí nghiệm, hoặc mở rộng thành nền tảng cho các ứng dụng nhà thông minh và nông nghiệp thông minh.
- Về mặt định hướng tương lai: Đây là bước khởi đầu để nghiên cứu và phát triển các hệ thống IoT lớn hơn, tích hợp nhiều loại cảm biến khác nhau, hỗ trợ trí tuệ nhân tạo để phân tích dữ liệu và dự đoán xu hướng.

CHƯƠNG II. THIẾT KẾ HỆ THỐNG

2.1. Phân tích yêu cầu hệ thống

2.1.1. Phân tích yêu cầu hệ thống

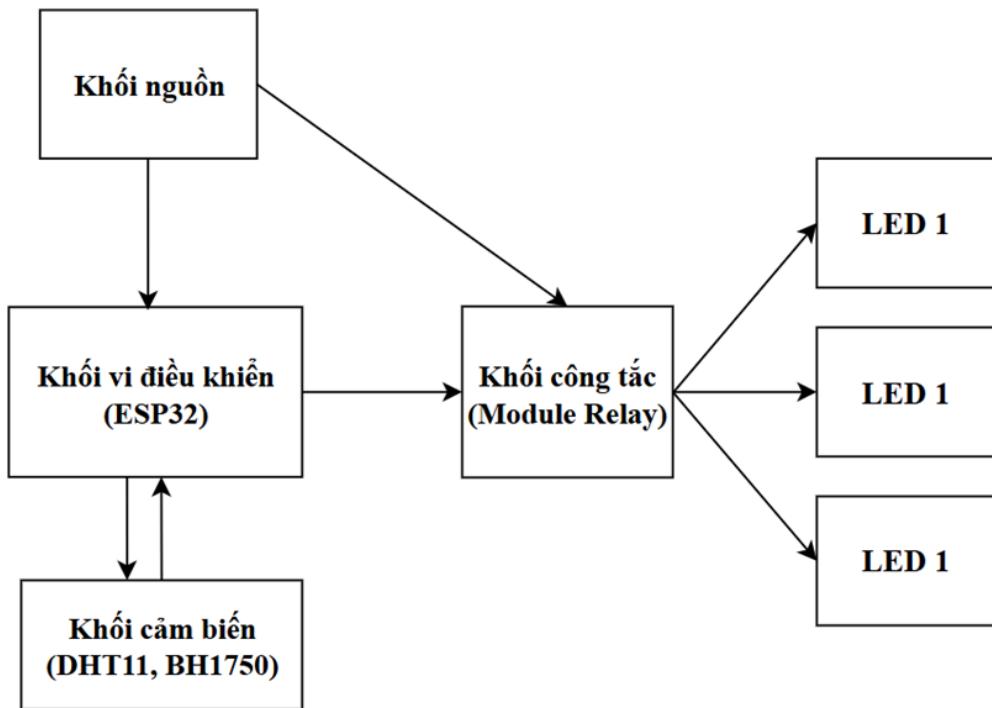
- Thu thập dữ liệu cảm biến (nhiệt độ, độ ẩm, ánh sáng).
- Truyền dữ liệu về máy chủ qua giao thức MQTT.
- Lưu trữ dữ liệu vào cơ sở dữ liệu.
- **Hiển thị dữ liệu theo thời gian thực** (bảng, biểu đồ).

2.1.2. Yêu cầu chức năng

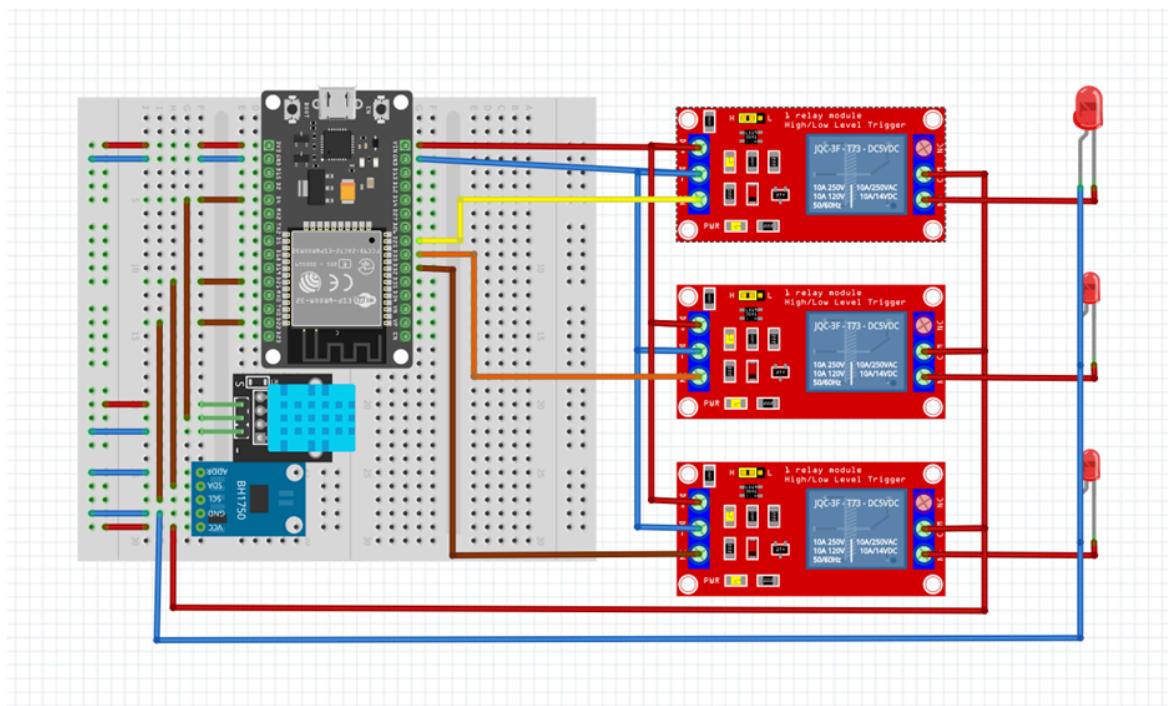
- Hệ thống phải hoạt động ổn định, liên tục.
- Dữ liệu được cập nhật theo thời gian thực.
- Khả năng mở rộng khi thêm nhiều cảm biến.
- Giao diện dễ sử dụng, trực quan

2.2. Thiết kế phần cứng.

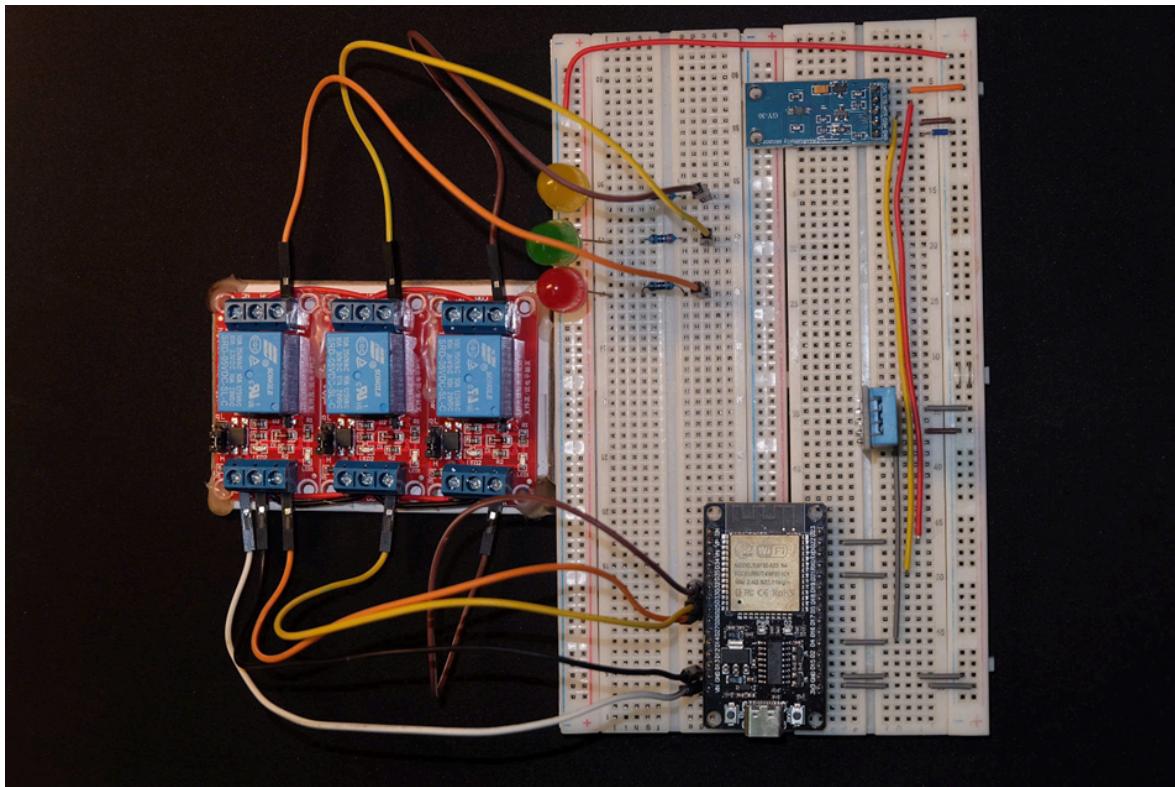
2.2.1. Sơ đồ khái



2.2.2. Sơ đồ mạch



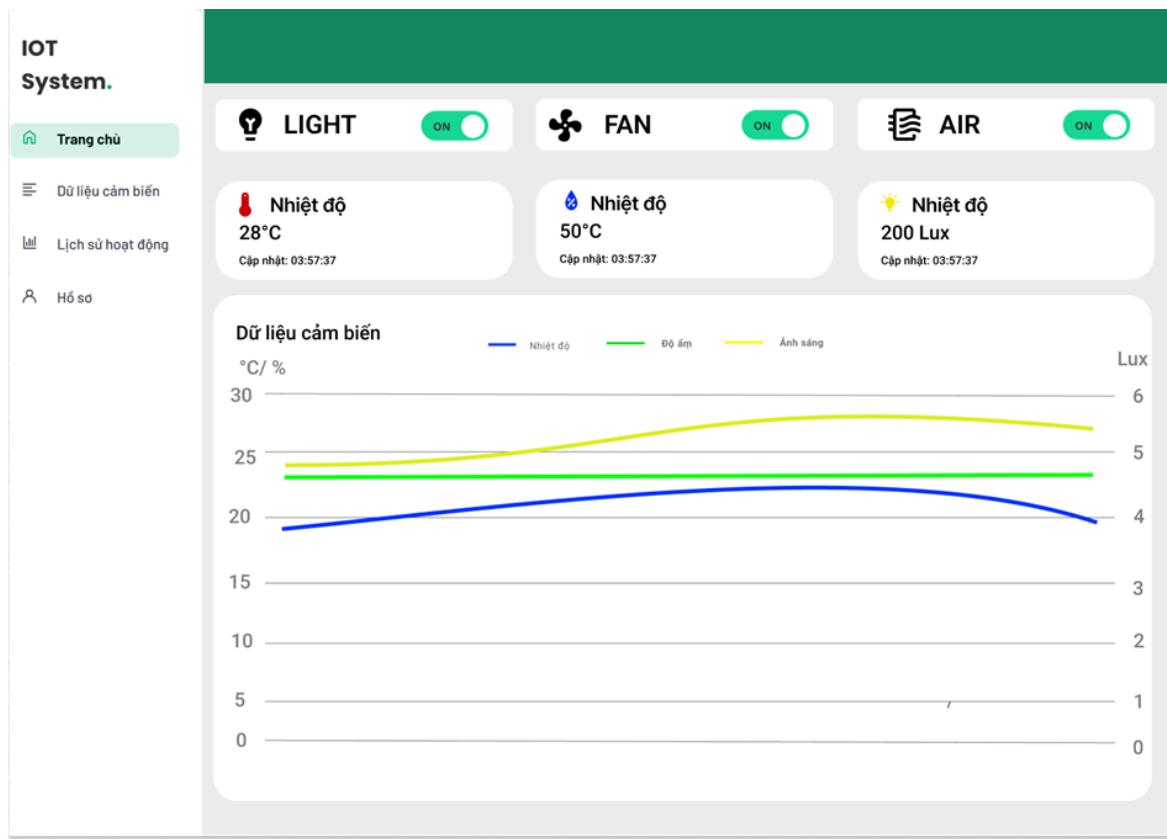
2.2.3. Sơ đồ thực tế



2.3. Thiết kế giao diện

2.3.1. Giao diện trang chủ

- Khu vực điều khiển thiết bị, có 3 ô hiển thị trạng thái thiết bị:
 - LIGHT (bóng đèn): kèm icon bóng đèn, công tắc ON màu xanh.
 - FAN (quạt): kèm icon quạt gió, công tắc ON màu xanh.
 - AIR (điều hòa): kèm icon điều hòa, công tắc ON màu xanh.
- Khu vực cảm biến: Hiển thị thông số của cảm biến (nhiệt độ, độ ẩm, ánh sáng), giá trị cảm biến đo được và thời gian đo theo dạng HH:mm:ss
- Phần biểu đồ đường hiển thị 3 thông số: nhiệt độ, độ ẩm, ánh sáng. Trục tung bên trái hiển thị giá trị °C / %, trục tung bên phải hiển thị đơn vị Lux.



2.3.2. Giao diện dữ liệu cảm biến

- Thanh tìm kiếm và bộ lọc:
 - Ô nhập tìm kiếm: cho phép gõ từ khóa để tìm dữ liệu cảm biến theo tiêu chí cụ thể.
 - Nút lọc “Mới nhất/ Cũ nhất”: giúp hiển thị dữ liệu mới nhất/ cũ nhất được ghi nhận.
 - Nút lọc “Tất cả cảm biến/ Nhiệt độ / Độ ẩm/ Ánh sáng”: cho phép chọn loại cảm biến muốn hiển thị (ví dụ: nhiệt độ, độ ẩm, ánh sáng).
 - Nút “Tìm kiếm”: thực hiện truy vấn dữ liệu theo điều kiện đã nhập/chọn.
- Bảng dữ liệu cảm biến:
 - Cấu trúc bảng: số thứ tự(STT), nhiệt độ, độ ẩm, ánh sáng, thời gian (dd-MM-yyyy HH:mm:ss)
 - Các tính năng bảng: Xuất dữ liệu ra file CSV, tùy chọn số bản ghi hiển thị (ví dụ: 15 bản ghi/trang), phân trang.

STT	Nhiệt độ	Độ ẩm	Ánh sáng	Thời gian
1	28.5	28.5	28.5	2025-08-20 10:00:00
2	28.5	28.5	28.5	2025-08-20 10:00:00
3	28.5	28.5	28.5	2025-08-20 10:00:00
4	28.5	28.5	28.5	2025-08-20 10:00:00
5	28.5	28.5	28.5	2025-08-20 10:00:00
6	28.5	28.5	28.5	2025-08-20 10:00:00
7	28.5	28.5	28.5	2025-08-20 10:00:00
8	28.5	28.5	28.5	2025-08-20 10:00:00

Hiển thị 15 bản ghi (trang 1/331)

2.3.3. Giao diện lịch sử hoạt động

- Trang “Lịch sử hoạt động” được thiết kế để người dùng theo dõi và tra cứu toàn bộ hành động điều khiển thiết bị trong hệ thống IoT, với các tính năng tìm kiếm, lọc và xuất dữ liệu tiện lợi
- Thanh tìm kiếm và bộ lọc:
 - Ô nhập tìm kiếm: hỗ trợ tìm kiếm lịch sử hoạt động, theo tên thiết bị.
 - Nút lọc “Mới nhất/ Cũ nhất”: giúp hiển thị dữ liệu mới nhất/ cũ nhất được ghi nhận.
 - Nút lọc “Tất cả thiết bị”: cho phép chọn loại thiết bị (ví dụ LIGHT, FAN, AIR).
 - Nút lọc “Tất cả hành động”: chọn loại hành động (ON, OFF).
 - Nút “Tìm kiếm”: thực hiện truy vấn dữ liệu theo điều kiện đã nhập/chọn.

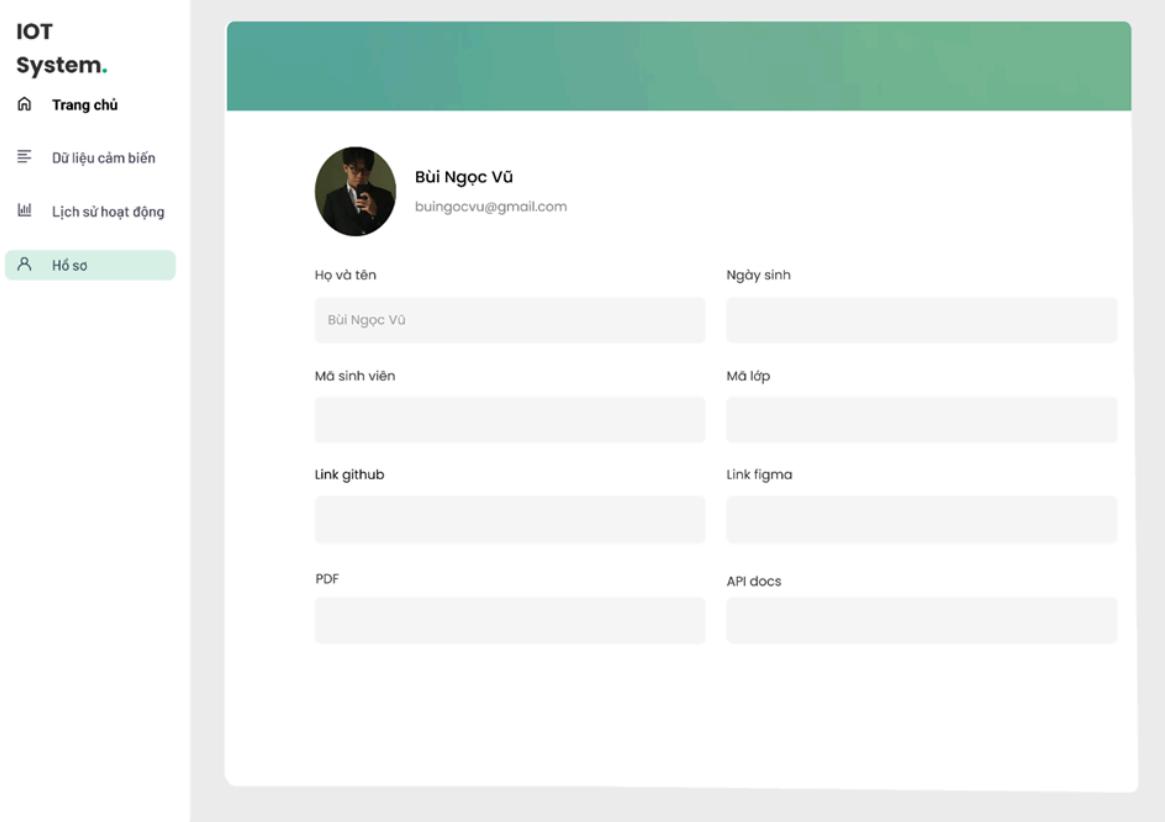
- Bảng lịch sử hoạt động:
 - Cấu trúc bảng: số thứ tự(STT), nhiệt độ, độ ẩm, ánh sáng, thời gian (dd-MM-yyyy HH:mm:ss)
 - Các tính năng bảng: Xuất dữ liệu ra file CSV, tùy chọn số bản ghi hiển thị (ví dụ: 15 bản ghi/trang), phân trang.

STT	Thiết bị	Hành động	Thời gian
1	LIGHT	ON	2025-08-20 10:00:00
2	FAN	OFF	2025-08-20 10:00:00
3	AIR	OFF	2025-08-20 10:00:00
4	LIGHT	OFF	2025-08-20 10:00:00
5	FAN	ON	2025-08-20 10:00:00
6	AIR	OFF	2025-08-20 10:00:00
7	FAN	ON	2025-08-20 10:00:00
8	AIR	ON	2025-08-20 10:00:00

Hiển thị 1-15 trong 70 bản ghi (trang 1/5)

2.3.4. Giao diện hồ sơ

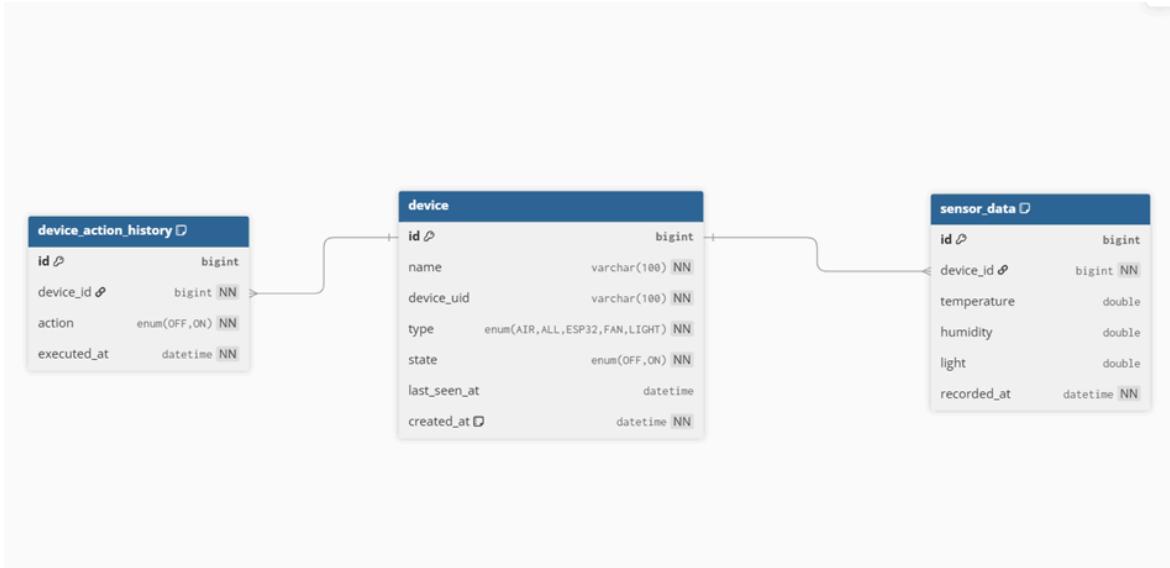
- Trang “Hồ sơ” trong hệ thống được thiết kế với bố cục gọn gàng, thể hiện đầy đủ các thông tin nhận diện và dữ liệu liên quan đến người dùng.
- Bên dưới là các trường thông tin được bố trí theo dạng biểu mẫu, bao gồm: họ và tên, ngày sinh, mã sinh viên, mã lớp, liên kết GitHub, liên kết Figma, tài liệu PDF và tài liệu API.



2.4. Thiết kế cơ sở dữ liệu

2.4.1. Lược đồ ERB

- device là bảng trung tâm, đóng vai trò định danh từng thiết bị trong hệ thống.
- device_action_history và sensor_data đều phụ thuộc vào device thông qua khóa ngoại.
- Sơ đồ quan hệ:
 - device → (1-n) → device_action_history
 - device → (1-n) → sensor_data



2.4.2. Mối quan hệ giữa các bảng

- Quan hệ giữa device và device_action_history:
- Bảng device là bảng trung tâm quản lý thông tin thiết bị IoT (đèn, quạt, máy lạnh, ESP32,...).
- Bảng device_action_history lưu lại lịch sử hành động bật/tắt của từng thiết bị.
- Mối quan hệ: 1-nhiều (one-to-many):
 - Mỗi thiết bị (device) có thể phát sinh nhiều hành động (device_action_history) trong quá trình vận hành.
 - Liên kết thông qua khóa ngoại device_id trong bảng device_action_history trả tới id của bảng device.

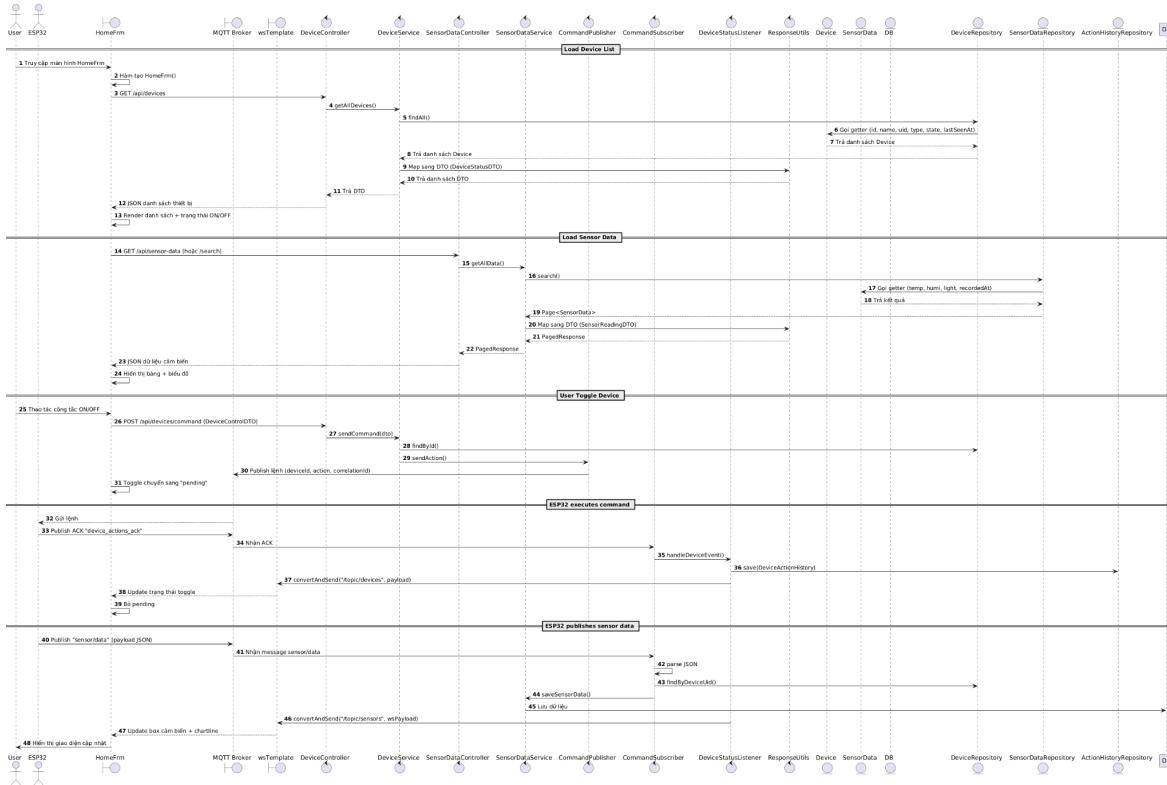
2.5. Sequence Diagram

2.5.1. Sequence 1: HomeFrm hiển thị thiết bị, dữ liệu cảm biến, điều khiển ON/OFF

- Diễn giải
 - Người dùng truy cập vào màn HomeFrm
 - Hàm tạo HomeFrm() được gọi
 - HomeFrm() gọi phương thức getAll() của DeviceController qua API GET /api/devices
 - Phương thức getAll() của DeviceController được gọi
 - Phương thức getAll() gọi đến DeviceService.getAllDevices()
 - Phương thức getAllDevices() được gọi
 - DeviceService gọi truy cập entity để lấy dữ liệu bằng deviceRepo.findAll()

8. Entity Device gọi các phương thức getter (getId(), getName(), getDeviceUid(), getType(), getState(), getLastSeenAt())
9. Entity Device trả danh sách đối tượng Device cho DeviceRepository rồi trả về cho DeviceService
10. DeviceService map sang DTO (DeviceStatusDTO) và trả kết quả cho DeviceController
11. DeviceController trả JSON danh sách thiết bị cho HomeFrm()
12. HomeFrm render danh sách thiết bị và đặt trạng thái ON/OFF cho từng công tắc
13. HomeFrm() gọi phương thức getAll() của SensorDataController qua API GET /api/sensor-data (hoặc gọi search() qua GET /api/sensor data/search khi có tham số lọc)
14. Phương thức getAll() của SensorDataController được gọi
15. SensorDataController gọi SensorDataService.getAllData()
16. Phương thức getAllData() của SensorDataController được tạo
17. getAllData() lấy dữ liệu từ entity SensorData bằng phương thức sensorRepo.search()
18. Entity SensorData gọi các phương thức getter (getTemperature(), getHumidity(), getLight(), getRecordedAt())
19. Entity SensorData trả trang đối tượng cho SensorDataRepository và trả Page cho Service
20. SensorDataService map sang DTO (SensorReadingDTO) và gói trong PagedResponse
21. SensorDataController trả JSON dữ liệu cảm biến cho HomeFrm()
22. HomeFrm cập nhật giá trị nhiệt độ/độ ẩm/ánh sáng và đổ dữ liệu ban đầu vào biểu đồ
23. Người dùng thao tác công tắc ON/OFF trên HomeFrm.
24. HomeFrm gọi API POST /api/devices/command tới DeviceController với payload DeviceControlDTO (deviceId, action).
25. Phương thức sendCommand() của DeviceController được gọi.
26. DeviceController gọi DeviceService.sendCommand(dto).
27. Phương thức sendCommand(dto) được gọi.
28. DeviceService gọi CommandPublisher bằng phương thức deviceRepo.findById() để kiểm tra thiết bị và publish lệnh tới MQTT broker.
29. Phương thức sendAction() của CommandPublisher được gọi
30. Phương thức sendAction() của CommandPublisher gửi thông điệp topic MQTT với payload(deviceId, action, correlationId)

- 31. HomeFrm chuyển toggle sang "pending".
- 32. MQTT Broker chuyển lệnh đến ESP32.
- 33. ESP32 publish ACK lên "device_actions_ack" với payload JSON
- 34. CommandSubscriber nhận ACK
- 35. CommandSubscriber gọi DeviceStatusListener
- 36. Phương thức handleDeviceEvent() của DeviceStatusListener được gọi.
- 37. DeviceStatusListener tạo DeviceActionHistory mới và lưu qua ActionHistoryRepository
- 38. DeviceStatusListener gọi wsTemplate.convertAndSend("/topic/devices", payload) để trả về HomeFrm
- 39. HomeFrm cập nhật trạng thái toggle và bỏ qua pending.
- 40. ESP32 publish dữ liệu lên MQTT topic "sensor/data" với payload JSON tới MQTT Broker
- 41. CommandSubscriber nhận message từ MQTT Broker
- 42. CommandSubscriber gọi tới CommandSubscriber với topic="sensor/data" và message=.
- 43. Phương thức handleDeviceEvent() parse JSON từ message và gọi handleSensor() để xử lý:
 - a. Đọc deviceUid, temperature, humidity, light/light_level từ payload
 - b. Xử lý giá trị (-1 được xem là sentinel → null)
 - c. Tìm Device theo deviceRepository.findByDeviceUid(deviceUid)
- 44. sensorDataService.saveSensorData() được gọi để lưu dữ liệu vào DB.
- 45. Phương thức wsTemplate.convertAndSend("/topic/sensors", wsPayload) được gọi
- 46. DeviceStatusListener đóng gói dữ liệu vào wsPayload để gửi realtime tới HomeFrm
- 47. HomeFrm cập nhật box cảm biến, chartline
- 48. HomeFrm hiển thị cho người dùng.
- Sequence

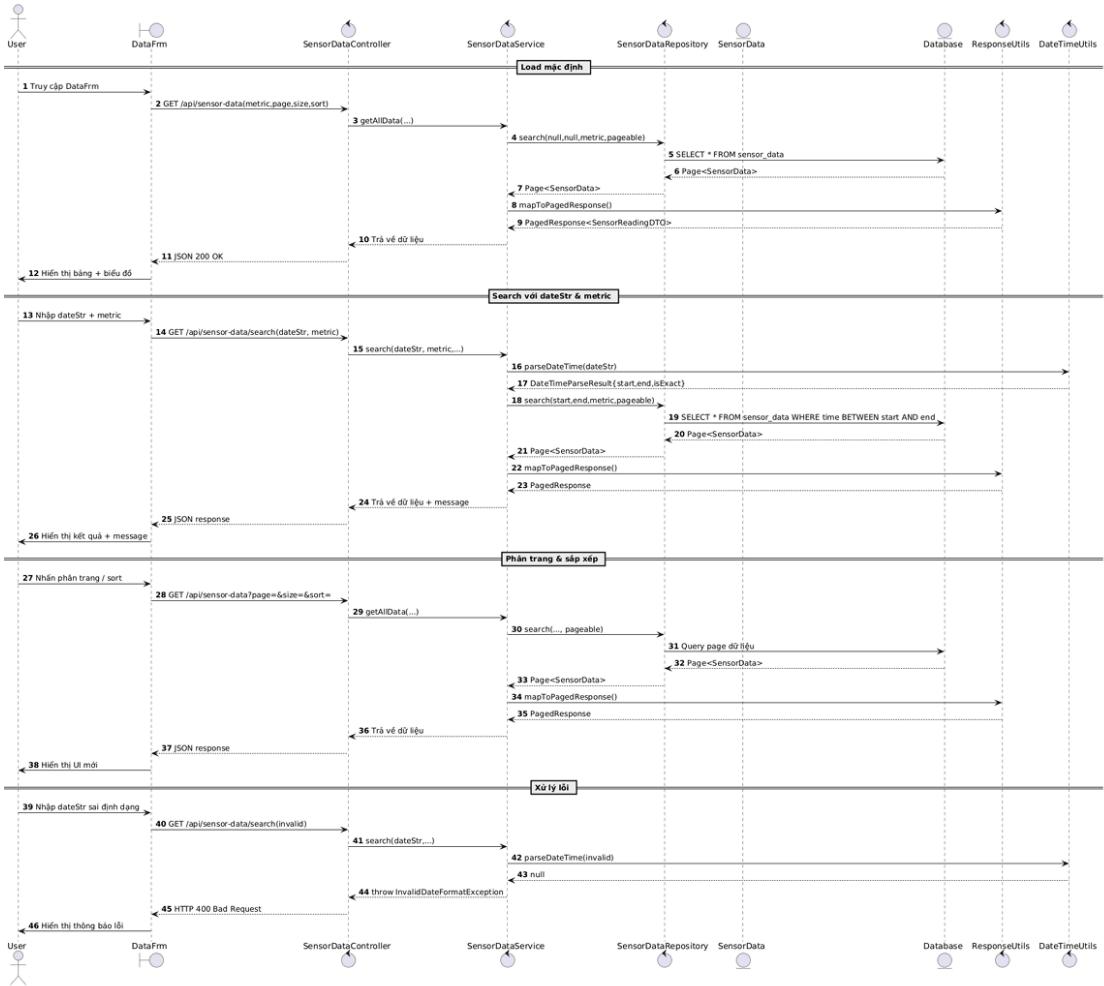


2.5.2. Sequence 2: Hiển thị dữ liệu cảm biến với tìm kiếm và lọc

- **Điễn giải**

1. Người dùng click menu "DỮ LIỆU CẢM BIẾN".
2. DataFrm (Boundary) được khởi tạo và hiển thị giao diện tìm kiếm mặc định.
3. DataFrm gọi API GET /api/sensor-data với tham số mặc định.
4. SensorDataController.getAll() (Control) nhận request.
5. SensorDataController gọi SensorDataService.getAllData(...).
6. SensorDataService tạo PageRequest.of(page, size, Sort.by("recordedAt")).
7. SensorDataService gọi SensorDataRepository.search(null, null, metric, pageable).
8. SensorDataRepository truy vấn Database.
9. Database trả về Page.
10. SensorDataRepository trả dữ liệu cho SensorDataService.
11. SensorDataService gọi ResponseUtils.mapToPagedResponse(Page).
12. ResponseUtils trả PagedResponse cho SensorDataService.
13. SensorDataService trả PagedResponse cho SensorDataController.
14. SensorDataController trả JSON (HTTP 200) cho DataFrm.

- 15. DataFrm hiển thị bảng và biểu đồ dữ liệu cảm biến.
- 16. Người dùng nhập dateStr và chọn metric → click "Tìm kiếm".
- 17. DataFrm gọi API GET /api/sensor-data/search?dateStr=...&metric=....
- 18. SensorDataController.search() nhận request và gọi
 SensorDataService.search(...).
- 19. SensorDataService gọi DateTimeUtils.parseDate(dateStr).
- 20. DateTimeUtils trả DateTimeParseResult{start, end, isExact} hoặc ném lỗi.
- 21. Nếu hợp lệ, SensorDataService tạo PageRequest và gọi
 SensorDataRepository.search(start,end,metric,pageable).
- 22. SensorDataRepository truy vấn Database.
- 23. Database trả về Page.
- 24. SensorDataRepository trả dữ liệu cho SensorDataService.
- 25. SensorDataService ánh xạ sang PagedResponse qua ResponseUtils.
- 26. SensorDataService trả PagedResponse cho SensorDataController.
- 27. SensorDataController trả JSON cho DataFrm.
- 28. DataFrm cập nhật bảng, biểu đồ và message kết quả.
- 29. Người dùng thay đổi page/size hoặc chọn metric khác.
- 30. DataFrm gửi request GET /api/sensor-data?page=&size=&metric=....
- 31. SensorDataController nhận request và gọi SensorDataService.getAllData(...).
- 32. SensorDataService tạo PageRequest mới và gọi
 SensorDataRepository.search(...).
- 33. SensorDataRepository truy vấn Database.
- 34. Database trả về Page.
- 35. SensorDataRepository trả dữ liệu cho SensorDataService.
- 36. SensorDataService ánh xạ qua ResponseUtils sang PagedResponse.
- 37. SensorDataService trả PagedResponse cho SensorDataController.
- 38. SensorDataController trả JSON cho DataFrm.
- 39. DataFrm hiển thị dữ liệu mới (bảng/biểu đồ).
- 40. Người dùng click tiêu đề cột để sắp xếp.
- 41. DataFrm gửi request với tham số sort=asc/desc.
- 42. SensorDataService tạo Sort.by("recordedAt").ascending()/descending().
- 43. SensorDataRepository thực hiện query với sort mới.
- 44. Database trả về kết quả.
- 45. Kết quả đi qua SensorDataRepository → SensorDataService →
 SensorDataController.
- 46. DataFrm cập nhật bảng/biểu đồ theo dữ liệu đã sắp xếp.
- Sequence



CHƯƠNG III. XÂY DỰNG HỆ THỐNG

3.1. Cài đặt Arduino

```
1 #include <Wire.h>
2 #include <BH1750.h>
3 #include "DHT.h"
4 #include <WiFi.h>
5 #include <WiFiClientSecure.h>
6 #include <PubSubClient.h>
7 #include <ArduinoJson.h>
8
```

- Wire.h: Thư viện giao tiếp I2C để kết nối với cảm biến BH1750
- BH1750.h: Thư viện điều khiển cảm biến ánh sáng BH1750
- DHT.h: Thư viện điều khiển cảm biến nhiệt độ và độ ẩm DHT11
- WiFi.h và WiFiClientSecure.h: Cho kết nối WiFi với bảo mật SSL/TLS
- PubSubClient.h: Hỗ trợ giao thức MQTT • ArduinoJson.h: Xử lý dữ liệu định dạng JSON

```
8
9 #define DHTPIN 4
10#define DHTTYPE DHT11
11#define RELAY_TEMP 25
12#define RELAY_LIGHT 33
13#define RELAY_HUMI 32
14
```

- DHTPIN: Chân kết nối với cảm biến DHT11 (GPIO4)
- RELAY_TEMP/LIGHT/HUMI: Chân điều khiển các relay cho nhiệt độ, ánh sáng và độ ẩm

```
14
15 const char* ssid = "VAN TUAN";
16 const char* password = "tuan24689";
17
18 const char* mqtt_server = "7c3aa4ee26624b92a6748300e938cd6b.s1.eu.hivemq.cloud";
19 const int mqtt_port = 8883;
20 const char* mqtt_user = "esp32";
21 const char* mqtt_pass = "Bnv2003@";
22
23 const char* TOPIC_ACTIONS = "device_actions";
24 const char* TOPIC_ACTIONS_ACK = "device_actions_ack";
25 const char* TOPIC_SENSOR = "sensor/data";
26 const char* TOPIC_WILL = "device/status";
27 const char* DEVICE_UID = "esp32-1";
28
```

- Cấu hình kết nối WiFi và MQTT
- Sử dụng HiveMQ Cloud làm MQTT broker với kết nối bảo mật (cổng 8883)
- Các topic MQTT được sử dụng trong hệ thống:
 - TOPIC_ACTIONS: Nhận lệnh điều khiển thiết bị
 - TOPIC_ACTIONS_ACK: Phản hồi sau khi thực hiện lệnh

- TOPIC_SENSOR: Gửi dữ liệu cảm biến
- TOPIC_WILL: Thông báo trạng thái thiết bị (online/offline)

```

28
29 WiFiClientSecure espClient;
30 PubSubClient client(espClient);
31
32 DHT dht(DHTPIN, DHTTYPE);
33 BH1750 lightMeter;
34
35 bool overrideTemp = false;
36 bool overrideHumi = false;
37 bool overrideLight = false;
38 unsigned long overrideTempAt = 0;
39 unsigned long overrideHumiAt = 0;
40 unsigned long overrideLightAt = 0;
41 const unsigned long overrideTimeoutMs = 60000;
42
43 unsigned long lastSampleAt = 0;
44 const unsigned long sampleEveryMs = 3000;
45

```

- Các biến để theo dõi trạng thái điều khiển thủ công (override)
- Sau 60 giây (60000ms), hệ thống sẽ tự động trở về chế độ tự động
- Quản lý thời gian lấy mẫu dữ liệu cảm biến (mỗi 3 giây)

```

46 void connectWiFi() {
47   WiFi.mode(WIFI_STA);
48   WiFi.setSleep(false);
49   WiFi.begin(ssid, password);
50   while (WiFi.status() != WL_CONNECTED) { delay(200); }
51 }
--
```

- Thiết lập ESP32 ở chế độ station (kết nối đến router)
- Tắt chế độ sleep để duy trì kết nối liên tục
- Chờ đến khi kết nối WiFi thành công

```

-- 53 void onMessage(char* topic, byte* payload, unsigned int length) {
54     StaticJsonDocument<256> doc;
55     if (deserializeJson(doc, payload, length)) return;
56
57     int deviceId = doc["deviceId"] | 0;
58     const char* action = doc["action"] | "";
59     const char* cid = doc["correlationId"] | "";
60     bool turnOn = strcmp(action, "ON") == 0;
61
62     if (deviceId == 1) { digitalWrite(RELAY_TEMP, turnOn ? HIGH : LOW); overrideTemp = true; overrideTempAt = millis(); }
63     if (deviceId == 2) { digitalWrite(RELAY_LIGHT, turnOn ? HIGH : LOW); overrideLight = true; overrideLightAt = millis(); }
64     if (deviceId == 3) { digitalWrite(RELAY_HUMI, turnOn ? HIGH : LOW); overrideHumi = true; overrideHumiAt = millis(); }
65
66     StaticJsonDocument<200> ack;
67     ack["deviceId"] = deviceId;
68     ack["state"] = action;
69     ack["correlationId"] = cid;
70     char buf[200];
71     size_t n = serializeJson(ack, buf, sizeof(buf));
72     client.publish(TOPIC_ACTIONS_ACK, (uint8_t*)buf, n, false);
73 }
74

```

- Xử lý tin nhắn điều khiển nhận từ server
- Parse JSON để lấy thông tin deviceId, action và correlationId
- Điều khiển relay tương ứng dựa trên deviceId và action
- Đánh dấu trạng thái override và thời điểm bắt đầu override
- Gửi phản hồi (acknowledgement) về server kèm trạng thái hiện tại

```

/* 74
75 void resubscribe() {
76     client.subscribe(TOPIC_ACTIONS);
77 }
78
79 void reconnectMQTT() {
80     String clientId = "ESP32-" + String((uint32_t)ESP.getEfuseMac(), HEX);
81     while (!client.connected()) {
82         client.connect(clientId.c_str(), mqtt_user, mqtt_pass, TOPIC_WILL, 1, true, "offline");
83         if (client.connected()) {
84             client.publish(TOPIC_WILL, "online", true);
85             resubscribe();
86         } else {
87             delay(1000);
88         }
89     }
90 }
91

```

- resubscribe(): Đăng ký lại các topic sau khi kết nối lại MQTT • reconnectMQTT():

Xử lý kết nối MQTT với:

 - ClientID duy nhất dựa trên địa chỉ MAC của ESP32
 - Thiết lập Last Will Testament (LWT) để thông báo khi thiết bị mất kết nối
 - Thông báo "online" khi kết nối thành công
 - Đăng ký lại các topic cần thiết

```

91 void autoRelay(float t, float h, float lux) {
92     if (overrideTemp && millis() - overrideTempAt > overrideTimeoutMs) overrideTemp = false;
93     if (overrideHumi && millis() - overrideHumiAt > overrideTimeoutMs) overrideHumi = false;
94     if (overrideLight && millis() - overrideLightAt > overrideTimeoutMs) overrideLight = false;
95
96     if (!overrideTemp) digitalWrite(RELAY_TEMP, (!isnan(t) && t < 25) ? HIGH : LOW);
97     if (!overrideHumi) digitalWrite(RELAY_HUMI, (!isnan(h) && h < 10) ? HIGH : LOW);
98     if (!overrideLight) digitalWrite(RELAY_LIGHT, (lux > 0 && lux < 10) ? HIGH : LOW);
99 }
100 }
```

- Kiểm tra thời gian override đã vượt quá thời gian tối đa (60 giây)
- Tự động điều khiển relay dựa trên dữ liệu cảm biến nếu không trong chế độ override:
 - Relay nhiệt độ bật khi nhiệt độ dưới 25°C
 - Relay độ ẩm bật khi độ ẩm dưới 10%
 - Relay ánh sáng bật khi ánh sáng dưới 10 lux

```

- 101
102 void setup() {
103     Serial.begin(115200);
104     pinMode(RELAY_TEMP, OUTPUT);
105     pinMode(RELAY_LIGHT, OUTPUT);
106     pinMode(RELAY_HUMI, OUTPUT);
107     digitalWrite(RELAY_TEMP, LOW);
108     digitalWrite(RELAY_LIGHT, LOW);
109     digitalWrite(RELAY_HUMI, LOW);
110
111     connectWiFi();
112     Wire.begin(21, 22);
113     lightMeter.begin(BH1750::CONTINUOUS_HIGH_RES_MODE);
114     dht.begin();
115
116     espClient.setInsecure();
117     client.setServer(mqtt_server, mqtt_port);
118     client.setCallback(onMessage);
119     client.setKeepAlive(30);
120     client.setSocketTimeout(60);
121     client.setBufferSize(512);
122 }
123 }
```

- Khởi tạo serial monitor với baudrate 115200
- Thiết lập chân GPIO cho các relay và đặt trạng thái ban đầu là OFF (LOW)
- Kết nối WiFi
- Khởi tạo I2C (SDA=21, SCL=22) và các cảm biến
- Cấu hình MQTT client:
 - Sử dụng chế độ không xác minh chứng chỉ SSL (setInsecure())
 - Đặt server và callback function
 - Cấu hình keep-alive, timeout và buffer size

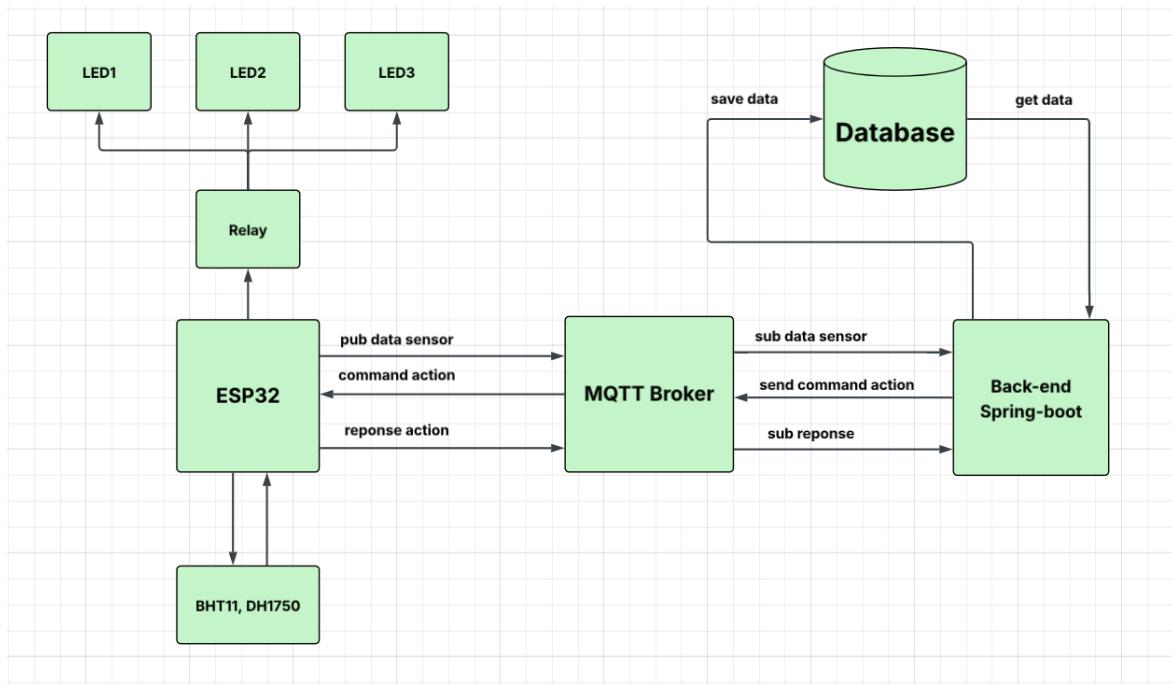
```

● 123
124 void loop() {
125     if (WiFi.status() != WL_CONNECTED) connectWiFi();
126     if (!client.connected()) reconnectMQTT();
127     client.loop();
128
129     unsigned long now = millis();
130     if (now - lastSampleAt >= sampleEveryMs) {
131         lastSampleAt = now;
132
133         float lux = lightMeter.readLightLevel();
134         float h = dht.readHumidity();
135         float t = dht.readTemperature();
136
137         StaticJsonDocument<256> doc;
138         doc["deviceUid"] = DEVICE_UID;
139         if (!isnan(t)) doc["temperature"] = t; else doc["temperature"] = nullptr;
140         if (!isnan(h)) doc["humidity"] = h; else doc["humidity"] = nullptr;
141         if (!isnan(lux)) doc["light"] = lux; else doc["light"] = nullptr;
142
143         char payload[256];
144         size_t n = serializeJson(doc, payload, sizeof(payload));
145         client.publish(TOPIC_SENSOR, (uint8_t*)payload, n, false);
146
147         autoRelay(t, h, lux);
148     }
149 }

```

- Đảm bảo kết nối WiFi và MQTT luôn duy trì
- Xử lý sự kiện MQTT thông qua client.loop()
- Định kỳ mỗi 3 giây:
 - Đọc dữ liệu từ các cảm biến
 - Tạo JSON chứa dữ liệu và ID thiết bị
 - Kiểm tra giá trị cảm biến hợp lệ trước khi gửi
 - Gửi dữ liệu đến MQTT broker
 - Cập nhật trạng thái relay dựa trên dữ liệu cảm biến

3.2. Thiết kế hệ thống

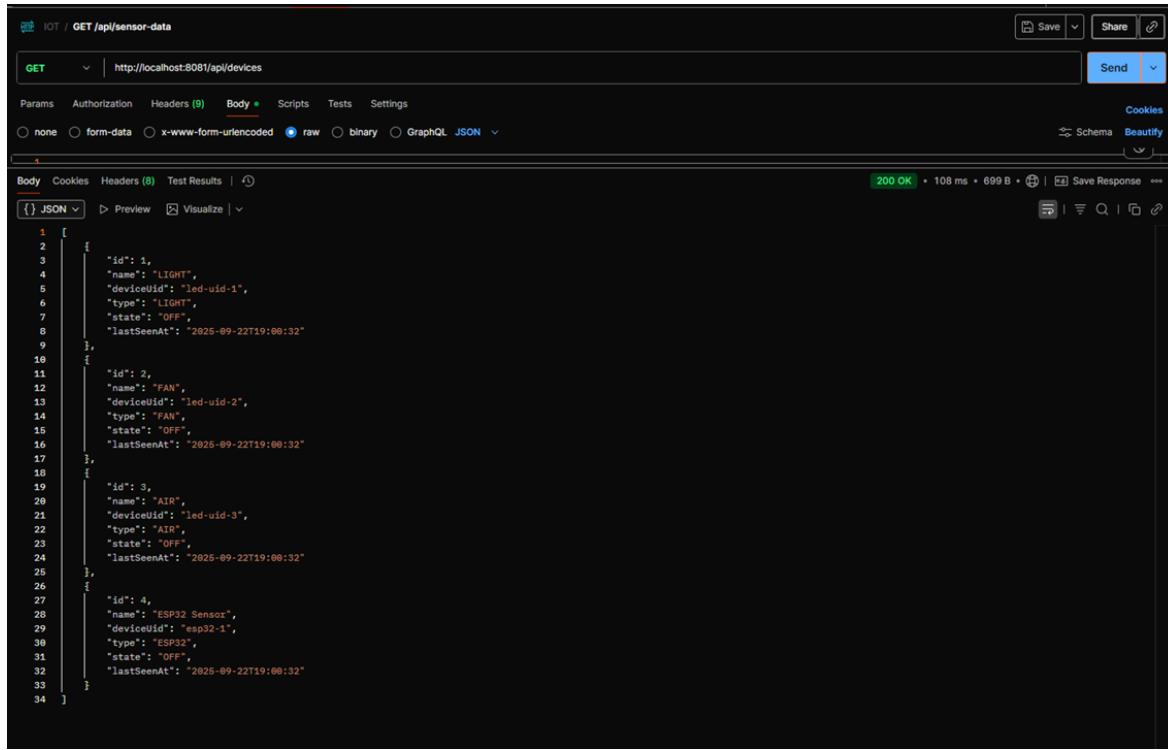


3.3. Cấu trúc hệ thống

```
iot-system/
├── src/main/java/com/iot_system/
│   ├── config/                      # Cấu hình (CORS, MQTT, WebSocket)
│   ├── controller/                 # REST Controllers
│   ├── domain/
│   │   ├── dto/                     # Data Transfer Objects
│   │   ├── entity/                  # JPA Entities
│   │   └── enums/                  # Enumerations
│   ├── exception/                 # Global Exception Handler
│   ├── mqtt/                       # MQTT Publisher/Subscriber
│   ├── repository/                # JPA Repositories
│   ├── service/                   # Business Logic
│   └── util/                      # Utility Classes
├── src/main/resources/
│   ├── static/                     # Frontend files
│   │   ├── css/                    # Stylesheets
│   │   ├── js/                     # JavaScript files
│   │   ├── img/                    # Images
│   │   └── *.html                  # HTML pages
│   └── application.properties      # Application config
└── target/                        # Maven build output
├── pom.xml                         # Maven dependencies
├── .env.example                     # Environment variables template
└── ENV_SETUP.md                    # Environment setup guide
└── README.md                        # This file
...
```

3.4. API

3.4.1. Lấy danh sách thiết bị



The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: <http://localhost:8081/api/devices>
- Body: raw JSON (selected)
- Response status: 200 OK
- Response time: 108 ms
- Response size: 699 B
- Response content (JSON):

```
[{"id": 1, "name": "LIGHT", "deviceId": "led-uid-1", "type": "LIGHT", "state": "OFF", "lastSeenAt": "2025-09-22T19:00:32"}, {"id": 2, "name": "FAN", "deviceId": "led-uid-2", "type": "FAN", "state": "OFF", "lastSeenAt": "2025-09-22T19:00:32"}, {"id": 3, "name": "AIR", "deviceId": "led-uid-3", "type": "AIR", "state": "OFF", "lastSeenAt": "2025-09-22T19:00:32"}, {"id": 4, "name": "ESP32 Sensoz", "deviceId": "esp32-1", "type": "ESP32", "state": "OFF", "lastSeenAt": "2025-09-22T19:00:32"}]
```

3.4.2. Gửi lệnh điều khiển

3.4.3. Tìm kiếm có điều kiện

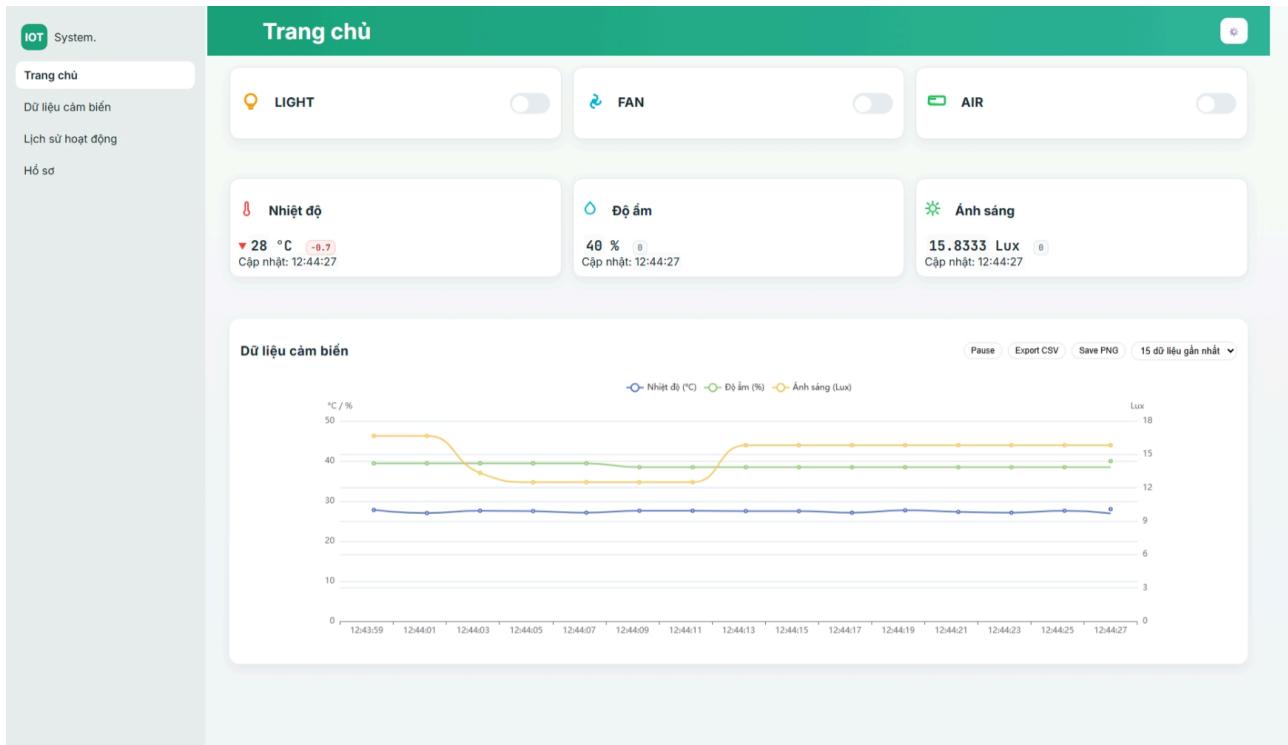
3.5. Giao diện

Thanh bên trái (Sidebar) màu xanh nhạt, gồm các mục:

- Trang chủ (đang được chọn, chữ màu đen, nền trắng).
- Dữ liệu cảm biến
- Lịch sử hoạt động
- Hồ sơ

3.5.1. Giao diện trang chủ

- Khu vực điều khiển thiết bị.Hiển thị ba công tắc điều khiển:
 - LIGHT (bóng đèn)
 - FAN (quạt)
 - AIR (máy lạnh/điều hòa)
- Tất cả đều đang tắt
- Khu vực hiển thị cảm biến: Nhiệt độ, độ ẩm, ánh sáng:
- Khu vực biểu đồ cảm biến
 - Biểu đồ dạng đường (line chart) thể hiện 3 thông số:
 - Nhiệt độ ($^{\circ}\text{C}$) – màu xanh dương.
 - Độ ẩm (%) – màu xanh lá.
 - Ánh sáng (Lux) – màu vàng.
 - Trục hoành: Thời gian (từng giây, ví dụ 12:43:59 → 12:44:27).
 - Trục tung trái: $^{\circ}\text{C} / \%$
 - Trục tung phải: Lux
 - Nút chức năng: Pause, Export CSV, Save PNG, chọn số lượng dữ liệu hiển thị (ví dụ: “15 dữ liệu gần nhất”).



3.5.2. Giao diện dữ liệu cảm biến

- Thanh công cụ tìm kiếm & lọc:
 - Ô tìm kiếm: "Nhập tìm kiếm...".
 - Bộ lọc bên phải:
 - Sắp xếp: Mới nhất, cũ nhất.
 - Cảm biến: Tất cả, nhiệt độ, độ ẩm, ánh sáng.
 - Hai nút chức năng:
 - Tìm kiếm (nút xanh lá).
 - Làm mới (nút viền xám, icon xoay vòng).
- Bảng dữ liệu: STT, nhiệt độ, độ ẩm, ánh sáng, thời gian
 - Nút CSV (xuất dữ liệu).
 - Bộ chọn hiển thị số lượng bản ghi
 - Phân trang

STT	Nhiệt độ (°C)	Độ ẩm (%)	Ánh sáng (Lux)	Thời gian
1	28.5	39.0	10.0	03-10-2025 13:12:41
2	28.7	39.0	10.0	03-10-2025 13:12:39
3	28.3	39.0	10.0	03-10-2025 13:12:37
4	28.4	39.0	10.0	03-10-2025 13:12:35
5	28.9	39.0	10.0	03-10-2025 13:12:33
6	28.1	39.0	10.0	03-10-2025 13:12:31
7	28.9	39.0	10.0	03-10-2025 13:12:29
8	28.0	39.0	12.5	03-10-2025 13:12:27
9	28.9	39.0	11.7	03-10-2025 13:12:25
10	28.2	39.0	10.0	03-10-2025 13:12:23
11	28.6	39.0	10.0	03-10-2025 13:12:21
12	28.0	39.0	10.0	03-10-2025 13:12:19
13	28.9	39.0	10.0	03-10-2025 13:12:17
14	28.0	39.0	10.0	03-10-2025 13:12:15
15	28.0	39.0	10.0	03-10-2025 13:12:13

3.5.3. Giao diện lịch sử hoạt động

- Thanh công cụ tìm kiếm & lọc:
 - Ô tìm kiếm: "Nhập tìm kiếm..."
 - Bộ lọc bên phải:
 - Sắp xếp: Mới nhất, cũ nhất.
 - Lọc theo hành động: Tất cả hành động, ON, OFF
 - Lọc theo thiết bị. (VD: LIGHT, FAN, AIR,...)
 - Hai nút chức năng:
 - Tìm kiếm (nút xanh lá).
 - Làm mới (nút viền xám, icon xoay vòng).
- Bảng dữ liệu: STT, tên thiết bị, hành động, thời gian thực hiện
 - Nút CSV (xuất dữ liệu).
 - Bộ chọn hiển thị số lượng bản ghi
 - Phân trang

The screenshot shows the 'Lịch sử hoạt động' (Activity History) page of the IoT System. The page has a green header with the title 'Lịch sử hoạt động'. Below the header is a search bar with placeholder 'Nhập tìm kiếm...', sorting options ('Sắp xếp: Mới nhất'), and filters ('Tất cả hành động', 'Tất cả thiết bị'). There are also buttons for 'Tim kiếm' (Search), 'Làm mới' (Reset), 'CSV', 'Hiển thị 15 bản ghi' (Display 15 records), and a 'Làm mới' (Reset) button.

The main content is a table with columns: STT (Index), Tên thiết bị (Device Name), Hành động (Action), and Thời gian thực hiện (Execution Time). The table contains 15 rows of data:

STT	Tên thiết bị	Hành động	Thời gian thực hiện
1	LIGHT	TẮT	28-09-2025 04:04:35
2	LIGHT	BẮT	28-09-2025 04:04:32
3	LIGHT	TẮT	28-09-2025 04:04:22
4	LIGHT	BẮT	28-09-2025 04:04:20
5	LIGHT	TẮT	28-09-2025 04:04:18
6	LIGHT	BẮT	28-09-2025 04:04:14
7	FAN	TẮT	28-09-2025 02:38:15
8	FAN	TẮT	28-09-2025 02:38:14
9	FAN	BẮT	28-09-2025 02:38:14
10	FAN	BẮT	28-09-2025 02:38:13
11	FAN	TẮT	28-09-2025 02:38:12
12	FAN	BẮT	28-09-2025 02:38:10
13	FAN	TẮT	28-09-2025 02:38:09
14	LIGHT	TẮT	28-09-2025 02:38:07
15	LIGHT	BẮT	28-09-2025 02:38:06

At the bottom, there is a message 'Hiển thị 1-15 trong 165 bản ghi (trang 1/11)' (Display 1-15 of 165 records (page 1/11)) and a navigation bar with buttons for '← Trước', 'Trang 1', '2', '3', '4', '5', 'Sau →'.

3.5.4. Giao diện hồ sơ

Nội dung hiển thị gồm: họ tên, ngày sinh, mã sinh viên, mã lớp, địa chỉ email, cùng các liên kết GitHub, Figma, API Docs và tệp PDF. Giao diện được bố trí rõ ràng, có biểu tượng minh họa và nút mở nhanh, bảo đảm thuận tiện trong việc tra cứu và truy cập tài liệu.

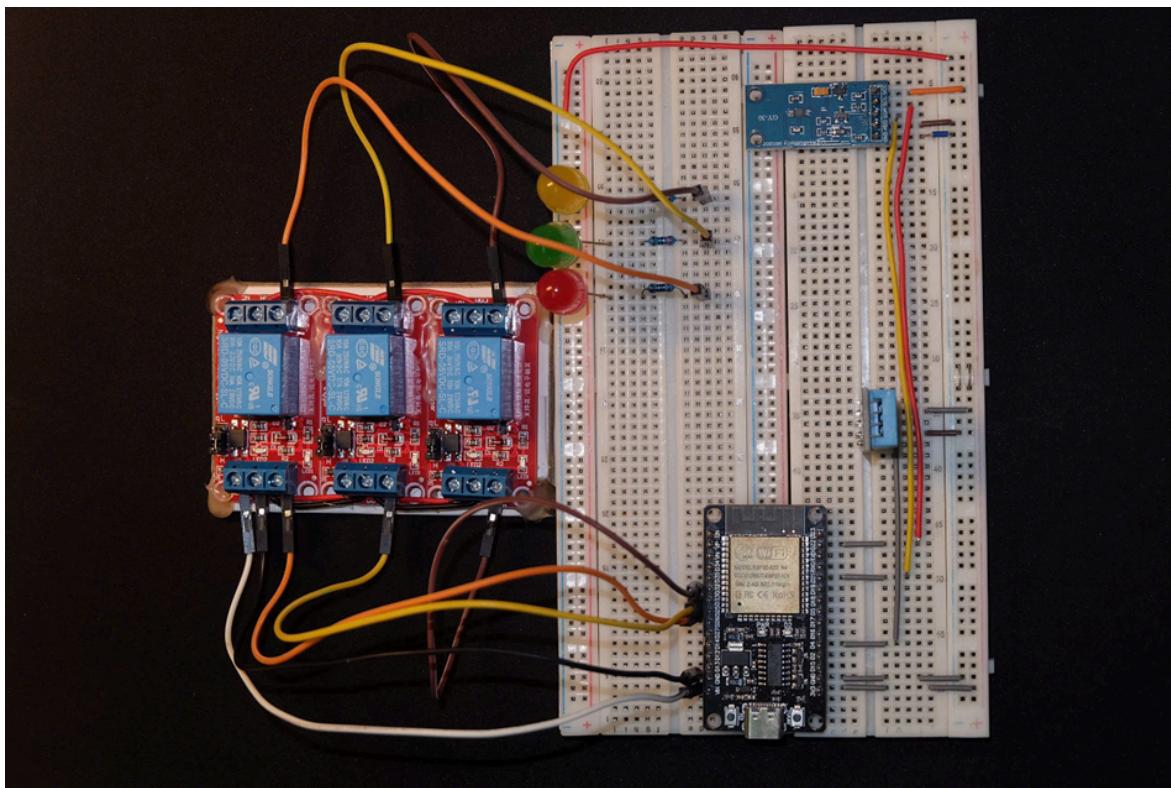
The screenshot shows a user profile page titled "Hồ sơ" (Profile) from the "IoT System". The profile belongs to "Bùi Ngọc Vũ" (Bui Ngoc Vu) with the email "buingocvuu@gmail.com". The page displays the following information:

Thông tin	Giá trị
Họ và tên	Bùi Ngọc Vũ
Ngày sinh	07/07/2003
Mã sinh viên	B22DCCN910
Mã lớp	D22HTTT05
GitHub	https://github.com/bngcvu/iot-system-smart
Figma	https://www.figma.com/design/lzKVYF7ZHbKTZa302cZMl/IOT-UNG-DUNG?node-id=...
API Docs	/swagger-ui.html
PDF	B22DCCN910.pdf

CHƯƠNG IV. KẾT LUẬN

4.1. Tổng quan

Trong quá trình phát triển hệ thống IoT, em đã thực hiện thành công các chức năng đo lường thời gian thực, điều khiển thiết bị điện từ xa, và lưu trữ dữ liệu cho người dùng. Hệ thống đã được triển khai và thử nghiệm, đáp ứng tốt các yêu cầu đề ra.



4.2. Cải tiến trong tương lai

Những cải tiến và mở rộng này sẽ giúp hệ thống trở nên mạnh mẽ, an toàn và linh hoạt hơn, đáp ứng được nhu cầu ngày càng tăng của người dùng và thích ứng với các xu hướng công nghệ mới trong lĩnh vực IoT và quản lý năng lượng thông minh:

- Tăng cường bảo mật
- Triển khai hệ thống xác thực và phân quyền cho người dùng.
- Cân nhắc sử dụng NoSQL (ví dụ: MongoDB) cho dữ liệu cảm biến nhằm tăng hiệu suất lưu trữ khối lượng dữ liệu lớn.
- Phát triển ứng dụng di động để người dùng dễ dàng theo dõi và điều khiển từ xa.
- Tích hợp trí tuệ nhân tạo để đưa ra các đề xuất tối ưu hóa năng lượng dựa trên dữ liệu cảm biến.

- Hỗ trợ thêm nhiều loại cảm biến và thiết bị IoT khác.
- Bổ sung công cụ phân tích dữ liệu nâng cao để cung cấp insights về xu hướng và mô hình tiêu thụ năng lượng.

LỜI CẢM ƠN

Em xin gửi lời cảm ơn sâu sắc đến thầy Nguyễn Quốc Uy, giảng viên môn học Internet of Things và Ứng dụng tại Học viện Công nghệ Bưu chính Viễn thông. Trong suốt khóa học, thầy đã truyền đạt kiến thức một cách tận tâm và chuyên nghiệp, giúp em và các bạn sinh viên hiểu sâu sắc về lĩnh vực IoT đang phát triển nhanh chóng. Những bài giảng sinh động và các ví dụ thực tế của thầy đã truyền cảm hứng cho chúng em, khơi dậy niềm đam mê với công nghệ và mở ra nhiều cơ hội mới trong tương lai nghề nghiệp. Thầy không chỉ là một người thầy giàu kinh nghiệm mà còn là một người cố vấn tận tụy. Sự hướng dẫn và ủng hộ của thầy trong quá trình thực hiện đồ án này đã giúp em vượt qua nhiều khó khăn, từ đó hoàn thiện dự án một cách tốt nhất. Kiến thức và kỹ năng em học được từ môn học này chắc chắn sẽ là nền tảng vững chắc cho sự phát triển nghề nghiệp của em trong tương lai. Em xin chân thành cảm ơn thầy vì tất cả những đóng góp quý báu này. Kính chúc thầy luôn mạnh khỏe, hạnh phúc và thành công trong sự nghiệp giảng dạy cao quý của mình.

Trân trọng,
Bùi Ngọc Vũ